A Multiplier-Free, Reduced Hessian Method for Process Optimization

Lorenz T. Biegler, Claudia Schmid, and David Ternet

EDRC 06-204-94

A MULTIPLIER-FREE, REDUCED HESSIAN METHOD FOR PROCESS OPTIMIZATION

Loreriz T. Biegler, Claudia Schmid, and David Ternet[1]

Process optimization problems typically consist of large systems of algebraic equations with relatively few degrees of freedom. For these problems the equation system is generally constructed by linking smaller submodels and solution of these models is frequently effected by calculation procedures that exploit their equation structure. In this paper we describe a tailored optimization strategy based on reduced Hessian Successive Quadratic Programming (SQP). In particular, this approach only requires Newton steps and their 'sensitivities' from structured process submodels and does not require the calculation of Lagrange multipliers for the equality constraints. It can also be extended to large-scale systems through the use of sparse matrix factorizations. The algorithm has the same superlinear and global properties as the reduced Hessian method developed in [4]. Here we summarize these properties and demonstrate the performance of the multiplier-free SQP method through numerical experiments.

1 Introduction.

Process optimization problems are encountered in design applications as well as in real time operations. These typically consist of large sets of nonlinear algebraic equations and represent the steady state operation of a chemical process. For optimization, these problems have relatively few degrees of freedom, as the number of design or control decision variables generally remains independent of the assumptions of the detailed phenomena in the process model. An example of a process optimization application is illustrated by the HDA process presented in Figure 1.

Solution of these nonlinear equations is generally covered by two types of simulation modes. In the *equation oriented mode* the model equations are

[1] **Chemical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213, email: biegler@cmu.edu**

collected within a single set and solved together by a general purpose solver, generally a large-scale Newton method. This approach tends to converge quickly but requires good initialization procedures and does not exploit the specific equation structure (e.g., block tridiagonal in distillation columns) of the process model. Several modeling systems have been developed that are based on this approach, including the SPEEDUP and ASCEND systems.

The second approach is a *modular mode* where equations for each process unit (cf. Figure 1) are solved separately and passed on to the next unit. As a result specialized solution procedures that exploit the unit structure can be used along with tailored initialization strategies. However, this approach converges slowly as the convergence of recycle streams requires an outer iteration loop and repeated solution of the unit models. For recycle convergence, a variety of fixed point algorithms have been applied and strategies that apply Newton or quasi-Newton methods to this problem are termed *simultaneous modular.* Despite its inefficiency, the modular mode remains the dominant process simulation mode in industry, as the simulation programs are easier to construct, convergence problems are localized within the individual submodels and these can often be corrected by physical intuition. ASPEN, PRO/II and HYSIM are the dominant commercial simulation packages in this class.

Process optimization problems take the form:

$$\min_{x \in R^n} f(x) \tag{1}$$

$$\text{subject to } c(x) = 0,\ x \in [x^L, x^V] \tag{2}$$

where $f : R^n \to R$ and $c : R^n \to R^m$ are generally assumed to be smooth functions. Here $n,m \gg n - m$ and the first derivatives of f and c are available, but their evaluation can be expensive in the modular mode. In this paper we consider algorithms that do not require second derivatives.

For both modes, the method of choice is Successive Quadratic Programming, although the NLP method is applied in different ways for each mode. The successive quadratic programming (SQP) method for solving (1)-(2) generates, at an iterate x_k, a search direction d by solving

$$\min_{d \in R^n} \nabla f(x_k)^T d + \frac{1}{2} d^T W(x_k) d \tag{3}$$

$$\text{subject to } c(x_k) + A(x_k)^T d = 0, \tag{4}$$

where g denotes the gradient of $f_y W$ denotes *tike* Hessian of the Lagrangian function $L(x, A) = f(x) + A^T c(z)$, ^nd this can be -approximated by a quasi-Newton update. A denotes the n xTOmatrix of constraint gradients

$$A(x) = [\nabla c_1(x), ..., \nabla c_m(x)]. \quad (5)$$

A new iterate is then computed as

$$x_{k+1} = x_k + \alpha_k d_k, \quad (6)$$

where a_k is a steplength parameter chosen so as to reduce the value of the merit function. In this study we will use the $l\backslash$ merit function

$$M^*) = f(*)+?M^*)\backslash\backslash u \quad (7)$$

where p is a penalty parameter; see for example [10], [16] or [13]. This penalty parameter is normally based on Lagrange multiplier values or their estimates. However, in this study, we consider a simpler measure that does not require Lagrange multiplier estimates, but still maintains descent properties for $4>n\{x)$.

For the *modular mode,* the NLP algorithm is applied to only a small part of the problem variables and most of the equations and variables are eliminated implicitly through solution of the models. As a result, the size of the optimization problem remains small (say, less than 100 variables) and no large scale extensions are required for SQP. On the other hand, because of nested solution of the process models and the need for reduced gradient information from these models (typically by finite difference), process optimization with this approach is less efficient than with the equation oriented mode.

For the *equation oriented* mode, on the other hand, a large NLP is considered directly by the SQP algorithm. All equations and variables are accessible and gradient calculations are straightforward. Nevertheless, care must be taken to solve these problems with a large-scale implementation of SQP. For this purpose, both sparse full-space methods [21] and reduced Hessian variations [22] of SQP have been proposed and demonstrated for process applications.

This study considers a variation of reduced Hessian SQP for a new formulation of process simulation problems. Here we seek to combine the best features of both simulation modes into a *tailored mode.* With this approach, we retain the modular structure for process model, along with any specialized safeguards and initialization features, but do not converge these

Newton-based process models in an inner loop. Instead a single Newton iteration is taken within each model. By collecting the Newton steps from each of the submodels and by obtaining additional projected gradient information from them, the overall effort for optimization can be made to be about the same as the work required for the equation oriented formulation.

However, as we assume the model equations, variables and their derivatives are not accessible directly, we find it necessary to develop a nonlinear programming method that requires neither second derivatives nor calculates Lagrange multiplier estimates for the model equations. In the next section we derive the 'multiplier free' reduced Hessian algorithm and present it formally for problem (1). Section 3 summarizes the convergence properties of the multiplier free method. In particular, the algorithm retains the global properties and the 1-step superlinear convergence properties derived for the reduced Hessian SQP method described in [4]. Section 4 describes two sets of numerical experiments. First, a comparison is given between the multiplier-free method and the reduced Hessian method described in [4]. Second, the results of a recent process optimization study [23] are summarized in order to demonstrate the tailored optimization strategy. Because of the multiplier-free method, a simple construction df the tailored approach is realized and it is demonstrated to perform as efficiently as the equation-oriented mode. The last section summarizes the paper and ties together some related work.

2 Derivation of Multiplier Free Algorithm

This section develops the expressions for the multiplier-free algorithm, which is a minor modification of the method developed in [4]. A key point in this method is that the multiplier estimates A need not be calculated explicitly and this property is useful for the tailored approach where direct access to the constraint normals (i.e., the Jacobians of the process models) may be difficult. The derivation is first developed for the equality constrained problem alone and convergence properties are summarized in the next section for this case. The addition of variable bounds is also considered briefly at the end of this section.

If we remove the variable bounds on (1), the solution of the quadratic program (3)-(4) can be written in a simple reduced form. We introduce a nonsingular matrix of dimension n, which we write as

$$[YkZ_k], \tag{8}$$

where Y_k 6 $R^{n\,x\,m}$ an<i Z_k 6 R^{nx}(n~fn\ and assume that

$$4\ Z_k = 0. \qquad 19)$$

(Prom now on we denote $A(x_k)$ as A_k, $g(x_k)$ as #*, etc.) Thus Z_k is a basis for the tangent space of the constraints. We express <&, the solution to (3)-(4),as

$$dk = Y_k p_Y + Z_k p_z, \qquad (10)$$

for some vectors py € R^m and p_z € R*"'*™. Due to (9) the linear constraints (4) become

$$c_k + AlY_k p_Y = Q. \qquad (11)$$

If we assume that A_k has full column rank then the nonsingularity of $[Y_k\ Z_k]$ and equation (9) imply that the matrix $A\wedge Y_k$ is nonsingular, so that p_Y is determined by (11):

$$p_Y = \sim[AlY_k]\text{-}^l c_k. \qquad (12)$$

The quadratic programming (QP) problem can be expressed exclusively in terms of the variables p_z- Substituting (10) into (3) with $Y_k p_Y$ determined, we obtain the unconstrained QP:

$$\min_{p_z \in R^{n-m}} (Z_k^T g_k + Z_k^T W_k Y_k p_Y)^T p_z + \backslash p l^T \{ZlW_k Z_k) p_z. \qquad (13)$$

In this study, we approximate $Z][W_k Z_k$ by the positive definite BFGS formula and the solution of (13) is

$$p_z = -(ZlW_k Z_k)\wedge[Zlg_k + ZlW_k Y_k pY). \qquad (14)$$

For this study, we partition x into m basic or dependent variables (which we reorder to be the first m variables) and $n—m$ decision variables, we induce the partition

$$A(x)^T = lC(x)N(x))_t \qquad (15)$$

where the mxm basis matrix $C(x)$ is assumed to be nonsingular. We now define $Z(x)$ and $Y(x)$ to be

$$\mathbf{W\text{-}[\text{-}°f\wedge W]yw..[\{].} \qquad \mathbf{(.6,}$$

This choice is particularly popular [19], [13] and advantageous when $A(x)$ is large and sparse, because a sparse LU decomposition of $C\{x)$ can often be

computed efficiently. Also, as shown in [23] an efficient partitioned decomposition can also be derived from the projected gradients in the modular mode in order to obtain the basis matrices in (16).

In [4] the cross term $[\hat{Z}_k W_k Y_k]pY$ is approximated by a vector w_k,

$$[ZRWtYkipvHWk, \tag{17}$$

without computing the matrix $Z\%W_kY_k$. This allows the rate of convergence of the algorithm to be 1-step Q-superlinear. Thus, the null space step (14) of our algorithm will be given by

$$p^* = \sim(2([W_kZ_kr^l[2Zg_k + CkWk], \tag{18}$$

where $0< C^* \leq 1$ is a damping factor described in [4].

Here the cross term is approximated either by a finite difference estimate along *YkPv* or by a quasi-Newton method. The finite difference estimate yields more accurate correction terms but also requires an additional gradient calculation. In [4] it was shown that such steps are needed whenever $PY = O(d_k)$ and $p_z = O(d_k)$. Here we define

$$w_k = Z\{x_k + Y_{kPY})^T g(x_k + Y_k p_Y) - Z\backslash g_k. \tag{19}$$

After the step to the new iterate $x_k+\backslash$ has been taken, we define

$$\overline{w}_k = a_k w_k \tag{20}$$

These correction terms are substituted for the ones defined in [4].

$$w^* = Zl[VL\{x_k + Y_k p_{Yy} X_k) - VL(s^*, A^*)]. \tag{21}$$

$$\overline{w}_k = Zl[VL\{x_k + a_k Y_k p_Y, \text{ AJH-I}) - VL(x_{fc}, A_{fc+1})] \tag{22}$$

where the (first order) multiplier estimates were calculated from:

$$x_k = -pfiUl-^y^*. \tag{23}$$

For the quasi-Newton approximation to w_k and $\overline{w}_k$, the rectangular matrix $Z\%W_k$ is approximated by a matrix 5&, using Broyden's method. We then obtain w_k by multiplying this matrix by $Y_k py$, i.e., $w_k = Skfapy$. $S_k+\backslash$ is updated so that it satisfies the following secant relation:

$$S_{k+1}(x_{k+1} - x_k) = zZ+tffa+x) - Z\%g(x_k) \tag{24}$$

and this leads to the Broyden update formula:

$$\bar S_{k+1} = \bar S_k + \frac{(\bar y_k - \bar S_k \bar s_k)\bar s_k^T}{\bar s_k^T \bar s_k}, \tag{25}$$

where

$$\bar y_k = Z_{k+1}^T g(x_{k+1}) - Z_k^T g(x_k) \tag{26}$$

$$\bar s_k = x_{k+1} - x_k. \tag{27}$$

thus defining

$$w_k = S_k Y_k p_Y \quad \text{and} \quad \bar w_k = \alpha_k \bar S_{k+1} Y_k p_Y. \tag{28}$$

Similarly, for B_{k+1}, the quasi-Newton approximation to the reduced Hessian Z^TWZ, satisfies the secant relation:

$$B_{k+1} s_k = y_k, \tag{29}$$

where s_k and y_k are defined by $s_k = \alpha_k p_Z$, and

$$y_k = Z_{k+1}^T g(x_{k+1}) - Z_k^T g(x_k) - \bar w_k \tag{30}$$

with B_k updated by the BFGS formula (cf. [13])

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{31}$$

provided $s_k^T y_k$ is sufficiently positive. As a result, the null space step is computed from:

$$B_k p_Z = -(Z_k^T g_k + \zeta_k w_k). \tag{32}$$

As in [4], we apply a safeguard to make sure that the Broyden corrections w_k and $\bar w_k$ remain bounded. At the beginning of the algorithm we choose a positive constant Γ and define

$$w_k := \begin{cases} w_k & \text{if } \|w_k\| \le \frac{\Gamma}{\|p_Y\|^{1/2}}\|p_Y\| \\ w_k \frac{\Gamma\|p_Y\|^{1/2}}{\|w_k\|} & \text{otherwise.} \end{cases}$$

On the other hand, the correction $\bar w_k$ is safeguarded by choosing a sequence of positive numbers $\{\gamma_k\}$ such that $\sum_{k=1}^{\infty} \gamma_k < \infty$, and requiring:

$$\bar w_k := \begin{cases} \bar w_k & [illegible] \\ \bar w_k \frac{[illegible]}{\gamma_k \|\bar w_k\|} & \text{otherwise.} \end{cases} \tag{34}$$

Similarly, to maintain a bounded positive definite BFQS update for B_k we use the same update criterion from [4]:

Choose a constant $j_{f6} > 0$ and a sequence of positive numbers {7} such that* Egl_17it < 00 *(this is the same sequence {7*} as in (34)). We represent Ok by any quantity which is of the same order as the error \\xk — x_m\\, and, as in [4], we use the optimality condition (WZ^gkW +* J|cfc||^.

- ***If ϖ_k** is computed by Broyden's method, and if both s[yk > 0 and*

$$\text{Hpv I} \leq \wedge \text{ l } |p_z| \text{ ,} \tag{35}$$

hold at iteration k, then update the matrix Bk by means of the BFGS formula (31) with Sk and y given by (30). Otherwise, set Bk+i = Bk**

- *IfWkis computed by finite differences, and if both s£yk > 0 and*

$$\|p_v\| \leq \gamma_{td}\|p_z\|/\sigma I^{/2} \tag{36}$$

hold at iteration k, then update the matrix Bk by means of the BFGS formula (31) with 8k and yk given by (30). Otherwise, set $B_{k+1} = B_k$.

Now for the BFGS updates, we know from [7], that if s£y* is always sufficiently positive and the following conditions are satisfied:

$$\frac{y_k^T s_k}{s_k^T s_k} \geq rn > 0 \tag{37}$$

$$\frac{\|y_k\|^2}{Vk^s k} \leq \text{M.} \tag{38}$$

for all $k \geq 1$, $sk \neq 0$, then at least half of the iterates at which updating takes place satisfy:

$$\frac{sjBjSj}{\|s_j\| \|_{\&} \quad \|} > \tag{39}$$

$$\beta_2 \leq \frac{\|B_j s_j\|}{\|s_j\|} \leq \beta_3 \tag{40}$$

and therefore remain bounded and uniformly positive definite. Since we need to refer to well-defined BFGS updates, we make the following definition.

Definition 2.1 *Let J be the set of iterates for which BFGS updating takes place and for which (39) arid (40) hold. We call J the set of "good iterates", and define* J* = *J D* {1,2,..., *k}.*

2,1 A Muitiplier-frfce Approach for Choosing jjL_k.

For the exact penalty line search, we choose \x so that for some $p > 0$,

$$/i||c(a:)||\geq|A(x)^{T}c(a:)|+p||c(x)|i \tag{41}$$

and show that this approach ensures a descent direction for the merit function. Moreover, for the good iterates J, it is a direction of strong descent.

Since d_k satisfies the linearized constraint (11) it is easy to show (see eq. (2.24) of [8]) that the directional derivative of the $t\$ merit function in the direction d_k is given by

$$D\phi_{\mu_k}(x_k\backslash d_k) = gld_k - \text{WbllCfcdli}. \tag{42}$$

Also, the fact that the same right inverse of $A(x)^T$ is used in (12) and (23) implies that

$$g(x)^{T}nx)py = X(xfc(x). \tag{43}$$

As a result of this relation, we show the following property:

Theorem 2.1 *Assume that A(x) is of full column rank for all x € D and that Z(x) is norm bounded. If <f>n(x) is defined by (7) and n satisfies (41) for all x € D, then Z?<^M(x; d) ≥ 0 for all d satisfying c{x) + A(x)^T d = 0 if and only if x is a Kuhn-Tucker point*

Proof. The proof is similar to the one in [13] for <^(x) with $\backslash i > \|A(\bar{x})\|_{oo}$. The *if part* follows from:

$$\begin{aligned} 0 \leq D\phi_{\mu}(\bar{x};d) &= g(\bar{x})^{T}d - \mu\|c(\bar{x})\|_1 \\ &= g(\bar{x})^{T}Z(\bar{x})p_z - \mu\|c(\bar{x})\|_1 + \lambda(\bar{x})^{T}c(\bar{x}) \\ &\leq g(x)^{T}Z(x)p_z\text{-}p\backslash\backslash c(x)\backslash\backslash i \\ &\leq g(\bar{x})^{T}Z(\bar{x})p_z \end{aligned} \tag{44}$$

for all p_z € $n^{n\sim m}$. This implies $Z(x)^{T}g(x) = 0$. Now if, in addition, $c(x) = 0$ then x is a Kuhn-Tucker point. If we assume $c(\bar{x}) \wedge 0$ then we can show the contradiction:

$$\begin{aligned} 0 \leq D\wedge\&d) &= X(x)^{T}c(x) - /\mathbf{xl}|\mathbf{o}(\wedge)|\mathbf{U} \cdot \\ &\leq -P\backslash\backslash c(\tilde{x})\backslash\backslash i \\ &< 0. \end{aligned} \tag{45}$$

The *only* i/ part follows from substitution of the Kuhn-Tucker conditions:

$$Z(\bar{x})^T g(x) = 0$$
$$c(\bar{x}) = 0 \quad (46)$$

into the directional derivative:

$$D\phi_\mu(\bar{x};d) = \bar{g}(x)^T \bar{Z}(x) p_z - fi\|c(\bar{x})\|i$$
$$= 0. \quad (47)$$

□

To show strong descent directions for good iterates, we recall the decomposition (32) and use (43) to obtain

$$D<t>v_k\{x_k|d_k) = glZkPz - n_k\|c_k\|i + \lambda_k^T c_k$$
$$= \{Zlg_k + Cwkfpz - Ckwjpz - /xjfdkdeHi + Aj[c_{fc}. \quad (48)$$

Now from (32) we have that

$$B_k s_k = -a_k(Z_k^r g_k + <;_k w_k). \quad (49)$$

If we satisfy the following property for n_k:

$$/x^*l|c^*|| \geq \wedge\wedge 1 + 2\wedge 11\wedge 11 \quad (50)$$

or, equivalently, from (43):

$$/i^*l|c^*|| \geq |\wedge r_k py| + 2p\|c_{fc}\| \quad (51)$$

then following the analysis on pp. 326-327 of [4] leads to;

$$D\phi_{\mu_j}(\wedge;di) \leq -\wedge U\wedge ill^2 - plkilli \quad (52)$$

for all j *e* J.

Now the penalty parameter /i* must satisfy (41), so we define it at every iteration of the algorithm by

$$"' = II \quad \mathbb{M} + 3, \quad \text{otherwise.} \quad <«*>$$

Note that for $c_k = 0, fi_k$ = /ifc-i and thus $/i_k$ is only updated when $c_k \wedge 0$.

Finally, the damping parameter (* is chosen so that the merit function retains the descent property even for large values of it;*. This is detailed in the algorithm presented next. In [4] it is shown that this damping parameter goes to unity in a neighborhood of the solution. As a. result, the choice of /i* and Ob ensure that strong descent properties hold for the good iterates *J.* For the other iterates, descent directions (though not necessarily of strong descent) can also be shown [4].

2.2 The Algorithm

Using the modifications of the reduced Hessian algorithm for the multiplier-free method, we now give a complete description of the algorithm. As in [5], the algorithm includes an approximation for the cross term using Broyden's method and finite differences, and is based on the relative sizes.of p_Y and j?$_z$. Calculation of the cross term and updating of the reduced Hessian proteed in a similar manner as in [5].

1. Set k := 1 and choose a starting point x\. Also, partition the variables into independent variables and dependent variables. Initialize the line search penalty parameter as /ii = 1. Initialize the Broyden matrix as S\ = [0 /]. Postmultiplying this matrix by *[Z Y]* shows that this initialization is in agreement with the initialization of the reduced Hessian matrix to B\ = *I.*

2. Evaluate /i, #i, c\ and A\ at a?i, and compute Y\ and Z\ as defined by (16).

3. Set *findiff = false* and compute p_Y by solving the system

$$\{A^Y_k)p_Y = -Cjfc. \qquad \text{(range space step)} \qquad (54)$$

4. Calculate w_k using Broyden's method, from equations (28) and (33).

5. Choose the damping parameter & from

$$C^* = \begin{cases} 1 & \text{if } g_k^T Z_k B_k Z_k^T w_k \geq 0 \\ \min\left(1, \dfrac{\beta\, g_k^T Z_k B_k Z_k^T w_k}{w_k^T Z_k B_k Z_k^T w_k}\right) & \text{otherwise} \end{cases} \qquad (55)$$

and compute p$_z$ from

$$B_k p_z = -[Z^T g_k + C_k W_k] \qquad (56)$$

6. Calculate a_k = ||Zjg*|| + ||c*|| and $\tilde{a}_k$ = max[||Zj0*||oo,||c^|oo]. If $\tilde{\sigma}_k \leq 0.1$,

$$\text{„ „ ^ 7/dllPzll} \qquad \text{,~x}$$
$$\text{IIPYII S } -\frac{}{\sigma_k\sqrt{2}}- \qquad \text{^°7'}$$

is satisfied and

$$\|p_v\| \leq \gamma_k^2 \|p_z\| \tag{58}$$

is not satisfied, set *findiff* = *true* and recompute w_k from

$$w_k = Z^T(x_k + Y_k py)g\{x_k + Y_{kPY}) - Z£g\{x_k) \tag{59}$$

(Note that we apply the finite difference update in a neighborhood of the solution, in order to avoid excessive gradient evaluations. We use d_k in this test, as it is less dependent on the problem dimension.)

7. If *findiff* = *true* use this new value of w_k to choose the damping parameter x_k from equation (55) and recompute p_z from equation (56).

8. Define the search direction by

$$d_k = Y_k py + Z_k Pz \tag{^0}$$

and set a_k = 1.

9. Test the line search condition

$$\text{^/xfci^ife} + ot_k d_k) \leq \text{<t>ii}_k(x_k) \text{ 4-O.lafcD^j^Xfcjdfc).} \tag{61}$$

10. If (61) is not satisfied, choose a new a* from

$$a_k = \max\left\langle \left[\frac{\text{-O.5D0}_{M_k}\ (x_k \backslash d_k)al_{\sim}}{\phi_{\mu_k}(x_k + \alpha_k d_k)^T - \phi_{\mu_k}(x_k) - \alpha_k D\phi_{\mu_k}(x_k; d_k)}, 0.1 \right] \right\rangle \tag{62}$$

and go to 9; otherwise set

$$x_{k+1} = x_k + a_k d_k \tag{63}$$

11. Evaluate /*+i, g_{k+x}, Cjt+i, A_{k+X} at z*+i, and compute Y_{k+i} and Z_{k+1}.

12. Update /x_{fc} to satisfy (53).

13. Update *Sk+i* using equations (25) to (27). If *findiff* = *false* calculate $\bar{W}k$ by Broyden's method through equations (28) and (34). Otherwise calculate $\bar{W}k$ by

$$\bar{W}k = <*kWk \tag{64}$$

and

$$\bar{w}_k = \begin{cases} \cdot_{1} & \text{if } \|\bar{w}_k\| \le \alpha_k \|p_Y\| / \bar{\gamma}_k \\ k \frac{\alpha_k \|p_Y\|}{\bar{\gamma}_k \|\bar{w}_k\|} & \text{otherwise} \end{cases} \tag{65}$$

Here, an upper bound on the finite difference correction term (weaker than the bound on the Broyden correction) is included (see Table 1.)

14. If $s\bar{\%}yk \le 0$ or if (57) is not satisfied, set J3*+i = £*. Else, compute

$$Sk = OL_k Pz \tag{66}$$

$$Vk = Zl+tfk+i-Zlgk-W\bar{k} \tag{67}$$

and update *Bk+* by the BFGS formula.

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T}{s_k^T B_k s_k} \wedge \quad \wedge \quad y_k s_k \tag{68}$$

15. Set A: = $k + 1$, and go to 3.

The numerical values for the parameters used to obtain the results in Section 4 are given in Table 1 and are the same ones used for the numerical comparison in [5]. Here *nind* is the number of independent variables of the problem and *k* is the iteration count. Numerical testing suggests that the values in Table 1 are reasonable for the examples considered in this paper.

Parameter	Reference	Suggested Value
r	33	20
7/rf	57	10
7*	58	$\frac{0.1nind^{0.25}}{(k)^{1.1}}$
7*	65	$\frac{0.01nind^{0.25}}{(k)^{1.1}}$

Table 1: Suggested value for parameters in Algorithm.

Finally, the algorithm terminates when the Kuhn-Tucker error falls below the user-specified tolerance. Again, we prefer ã* to *Ok* for this value as it is less dependent on problem size.

2.3 Extension to Bound Constrained Problems

To handle the bound constrained problem (1) we consider the decomposition in (10) and (11), and compute the nullspace step p_z through the solution of a low dimensional quadratic programming problem. Because n — m is small, solution of the QP problem is inexpensive and the modification is relatively easy to implement. For this problem we have adapted a version of the Goldfarb and Idnani QP algorithm [14}. A Complete description of the QP algorithm and its integration with the reduced Hessian SQP method is given in [22].

To solve for p_z the reduced dimension QP problem is given by:

$$\min_{\xi, p_z \in \mathbf{R}^{n-m}} \{Z^T_{k9k} + (_k w_k f p_z + \backslash_{Pz}{}^T (B_k) p_z + M\mathbf{ft} + \pounds^2/2) \qquad (69)$$

$$\text{subject to } x_k + (1 - OYkPy + Z\&z \in [z^L, z^u], \pounds \geq 0 \qquad (70)$$

The scalar variable £ is added to ensure that the QP always has a solution and the resulting search direction d_k keeps Xfc+i between bounds. This variable remains zero except for inconsistent constraint linearizations, normally at the initial stages of the optimization. Note that the QP algorithm requires a positive definite Hessian and this accounts for the quadratic term for£.

The QP (69) is substituted for the calculation of p_z in step 5 of the above algorithm.

In addition, the solution of the inequality constrained QP also leads to a minor modification in the calculation of $\backslash i_k$ by adding the multipliers, ν_k, calculated from the bound constraints in the QP. From the exact penalty function and the KKT conditions for the QP, we require that the penalty parameter in step 12 of the algorithm be chosen differently. Defining

$$\bar{9}^* = (1 - \xi)(g_k + \nu_k),$$

we require:

$$\mu_k = \begin{cases} \mu_{k-1} & \text{if } \mu_{k-1}\|c_k\|_1 \geq |\bar{g}_k^T Y_k p_Y| + 2\rho\|c_k\|_1 \\ \frac{|\bar{g}_k^T Y_k p_Y|}{\|c_k\|_1} + 3\rho & \text{otherwise.} \end{cases} \qquad (71)$$

instead of (53). To see this, we have for the directional derivative:

$$
\begin{aligned}
D\phi_\mu(x_k; d) &= g(x_k)^T d - \mu\|c(x_k)\|_1 \\
&= g_k^T Z_k p_z + (1-\xi) g_k^T Y_k p_Y - \mu\|c_k\|_1 \\
&= -(B_k p_z - \zeta_k w_k)^T p_z - \nu_k^T Z_k p_z + (1-\xi) \wedge\ Y_k p_Y - \mu\|c_k\|_1 \\
&= -(B_k p_z - \zeta_k w_k)^T p_z - \nu_k^T d_k + (1-\xi)(g_k + \nu_k)^T Y_k p_Y - \mu\|c_k\|_1 \\
&\leq -(B_k p_z - \zeta_k w_k)^T p_z + (1-\xi)(g_k + \nu_k)^T Y_k p_Y - \mu\|c_k\|_1 \\
&\leq -(B_k p_z - \zeta_k w_k)^T p_z - \rho\|c_k\|_1 \\
&< 0
\end{aligned}
$$

where the third relation follows from the KKT conditions of the QP, the fourth relation follows from (10), and the fifth relation follows from the sign of the multipliers *Vk* and the fact that z* is always between bounds. As a result, we still preserve the descent property and from (52) we have the strong descent property for all j € J.

The bound constraint modifications for this algorithm will be illustrated in the process optimization study in section 4.2

3 Summary of Convergence Properties

In this section we summarize several convergence results for equality constrained problems solved by the above algorithm. The proofs of these results will be sketched below and the interested reader is referred to [3] for a detailed analysis. Moreover, many of the results of [4] carry over directly or with only minor modifications.

We first show that the merit function <\>^ decreases significantly at the good iterates J, and that this gives the algorithms weak convergence property. To establish the main results we restate the following assumptions from [4].

Assumptions 3.1 The sequence *{xk}* generated by the algorithm is contained in a convex set *D* with the following properties.

(I) The functions / : R^n -> R and c : R^n -• R^m and their first and second derivatives are uniformly bounded in norm over *D.*

(II) The matrix *A(x)* has full column rank for all *x* € £>, and there exist constants, 70 and *fa* such that

$$\|Y(x)[A(x)^T Y(x))^{-1}\| \leq 70, \qquad \|Z(*)\| \leq /3b, \tag{72}$$

for all $x \in D$.

(III) For all $k \geq 1$ for which B_k is updated, (37) and (38) hold.

(IV) The correction term w_k is chosen so that there is a constant $K > 0$ such that for all k,

$$\|w^*\| \leq \|W^{1/a}\|. \tag{73}$$

The following theorem shows that the penalty parameter settles down and that the set of iterates is not bounded away from stationary points of the problem.

Theorem 3.1 *If Assumptions 3.1 hold, then the weights* {/i*} *are constant for all sufficiently large k and* \imirdk^oo{§Z% 9k\\ + ||cfc||) = 0.

Proof. First note that by Assumptions 3.1 (I)-(II) and (43) that

$$\{|g_k^T Y_k p_Y|/\|c_k\|_1\} = \{|\lambda_k^T c_k|/\|c_k\|_1\} \leq \{11^{**}11\} \tag{74}$$

is bounded. Therefore, since the procedure (53) increases /i* by at least p whenever it changes the penalty parameter, it follows that there is an index *ko* and a value /x such that for all A; > fco, /x* = *fi* such that p||cfc||i ≥ |A£cfc| + 2p||cjt||i. The rest of the proof follows in the same manner as in Lemma 4.1 and Theorem 4.2 in [4].

In [4] it was shown that if x* is a local minimizer that satisfies the second order optimality conditions, and if the penalty parameter /J* is chosen large enough, then x_m is a point of attraction for the sequence of iterates $\{x_k\}$ generated by the above algorithm. These are given as Assumption 5.2 and Lemmas 5.1, 5.2 and 5.3 in [4] and carry over to the above algorithm without modification. To prove these results the following assumptions were made in [4]. These assumptions are also necessary for the remaining analysis in this section.

Assumptions 3.2 The point x* is a local minimizer for problem (l)-(2) at which the following conditions hold.

(1) The functions / : R^n -• R and c : R^n -+ R^m axe twice continuously differentiable in a neighborhood of x,, and their Hessians are Lipschitz continuous in a neighborhood of z«.

(2) The matrix $A(x+)$ has full column rank. This implies that there exists a vector $A^* \in R^m$ such that

$$\nabla L(x_*, \lambda^* l = g\{x.\} + A(x^*)\backslash^* = 0.$$

(3) For all $q \in R^{n"m}$, $q \# 0$, we have $q^T Z?W^*Z.q > 0$.

(4) There exist constants 70, Po and j_c such that, for all x in a neighborhood of #•,

$$\|Y(x)[A(x)^r y(x)]^{-1}\| \leq 70, \quad \|Z(*)\| \leq A), \tag{75}$$

and

$$\|[y(x)Z(x)]^{-1}\| \leq_7 c. \tag{76}$$

(5) $Z(x)$ and A(x) are Lipschitz continuous in a neighborhood of z*, i.e. there exist constants y_z and 7^ such that •

$$\|\lambda(x) - \lambda(z)\| \leq \gamma_\lambda \|x - \wedge\|, \tag{77}$$

$$\backslash\backslash Z(x) - Z(z)\backslash\backslash \leq -yzWx\text{-}zU, \tag{78}$$

for all x, z near x*.

We can therefore assume that the iterates generated by the above algorithm converge to z«, which implies that for all large A; and some $p > 0$, $Hk = /i$ and

$$»\backslash\backslash c(x)\backslash\backslash \geq \sqcup (x)^T c(x)\backslash + p\backslash\backslash c(x)\backslash\backslash \tag{79}$$

in a neighborhood of x*. To complete the analysis we also consider when the BFGS updates are applied. We define U to be the set of iterates at which BFGS updating takes place,

$$U = \{* : B_{k+X} = BFGS(B_k, s_k, y_k)\}, \tag{80}$$

and let

$$U_k = UD \{1,2,..., A:). \tag{81}$$

The number of elements in U_k will be denoted by $\backslash U_k\backslash$. The following result from [4] carried over directly to the multiplier-free tnethod.

Theorem 3.2 *Suppose that the iterates* {x*} *generated by the multiplier free algorithm converge to a point x+ that satisfies Assumptions 3.2. Then for any k €If and any j ≥k*

$$\|x_i - x.\| < C\underline{r}\backslash^{u''}\backslash, \tag{82}$$

for some constants C > 0 and **0** ≤ r < **1.**

This result implies that if {|t/*|/fc} is bounded away from zero, then the multiplier free algorithm is R-linearly convergent. However, BFGS updating could take place only a finite number of times, in which case this ratio would converge to zero. It is also possible for BFGS updating to take place an infinite number of times, but every time less often, in such a way that \Uk\/k -> 0. Therefore the next result shows that the condition number of the matrices B_k is bounded, and that at the iterates *U* at which BFGS updating takes place the matrices B_k are accurate approximations of the reduced Hessian of the Lagrangian.

Theorem 3.3 *Suppose that the iterates* $\{x_k\}$ *generated by the above algorithm converge to a solution point x+ that satisfies Assumptions 3.2. Then* $\{\|B_k\|\}$ *and* $\{\|B^{\wedge'}\|\}$ *are bounded, and for allkeU*

$$MB_k - Z?W.Z.)p_z\| = o(\|d_k\|). \tag{83}$$

The proof follows along the same lines as the proofs of Lemma 5.5 and Theorem 5.6 in [4] with only slight modifications relating to differences in the definitions of y_k in (30). The entire proof is also redeveloped in [3] for the multiplier free case.

This result immediately implies that the iterates are R-linearly convergent, regardless of how often updating takes place.

Theorem 3.4 *Suppose that the iterates* $\{x_k\}$ *generated by the multiplier free algorithm converge to a solution point x+ that satisfies Assumptions 3.2. Then the rate of convergence is at least R-linear.*

Proof. Theorem 3.3 implies that the condition number of the matrices {2?*} is bounded. Therefore *all* the iterates are good iterates, and reasoning as in the proof of Theorem 5.4 we conclude that for all *j*

$$\|xj - x.\| \leq \underline{C}r^{>},$$

for some constants $C > 0$ and $0 \leq r < 1$.

We note that, as in Lemma 5.8 in [4], the Broyden matrices 5* also remain bounded and this follows directly from R-linear convergence and the well-known bounded deterioration property for Broyden's method (cf. Lemma 8.2.1 in [11]).

Finally, to establish 1-step superlinear convergence we need to assume that the steplengths a* have the value 1 for all large *k*. However the non-differentiable *l* merit function (7) used in this paper may reject steplengths of one, even though the lower bound on /i is weaker than $\|A\|_\infty^{-1}$ Thus the multiplier-free method can still suffer from the Maratos effect and the algorithm must be modified to allow unit steplengths and to achieve a fast rate of convergence. (In the numerical experiments described in the next section, we employ a non-monotone line search (or watchdog technique) of [9] that allows unit steplengths to be accepted for all large *k*. The analysis of the modified algorithm would be similar to that presented in §5.5 of [8])

Nevertheless, if we assume that the iterates generated by the above algorithm converge R-linearly to a solution and that unit steplengths are taken for all large *k*, then the performance of the method is no longer influenced by the merit function and the analysis is identical to that of [4]. The convergence result can therefore be summarized by:

Theorem 3.5 *Suppose that the iterates generated by the multiplier free algorithm converge R-linearly to a point x^ that satisfies Assumptions 3.2, and that* a* = 1 *for all large k. Then the rate of convergence is 1-step Q-superlinear.*

4 Numerical Experiments

The numerical experiments described in this section are divided into two parts. In the first part the multiplier free algorithm is compared with the algorithm analyzed and implemented in [4], [5] on a standard set of equality constrained test problems. It is shown that the multiplier free modifications perform well and generally lead to no loss of efficiency or reliability on these test problems. In the second part we summarize a process optimization case study presented in [23] for which the multiplier free method was developed. As in the first part, the multiplier free method performs well and therefore allows the use of existing process models, along with their own solution procedures, and without extensive reformulation of the model equations.

4.1 Equality Constrained Problems

In this subsection we consider a general collection of test problems; we include some smaller examples from Hodc and Schittkowski [17] as well as some scaleable problems from the GUTE set [6]. These test problems were also considered in [5] and the same tuning parameters (see Table 1) were used. In contrast to an extensive study of the correction terms in [5] we consider here only the complete algorithm and evaluate the effect of the multiplier free modifications. Also, as in [5] we also found it useful to scale the objective function; we arbitrarily choose an upper bound of 10 on $f(x_0)$.

Table 2 presents the results fin: problems taken from Hock and Schittkowski. MA28 was used to select the dependent variables using a threshhold tolerance of 1.0 in order to find a good pivot sequence, rather than minimize fill-in. Unless indicated otherwise and the convergence tolerance was set to 10^{-5}.

Problem	N/M	Multiplier Rree	With Multipliers
HS80	5/3	9(9/ 15/0.82)	10(10/ 15/0.81)
HS81	5/3	9(9/ 15/0.82)	10(10/ 15/0.81)
HS99	7/2	16(28/ 19/1.13)	16{ 19/ 17/6.84)
HS100	7/4	28(43/39/0.94)	28(43/ 39/0.95)
HS101	7/2	43(69/ 47/1.12)	53(85/ 56/1.16)
HS102	7/3	128(217/137/2.04)	44(70/ 49/1.11)
HS103	7/4	117(218/129/2.07)	57< 89/ 64/1.50)
HS104	8/4	29(70/ 39/1.25)	24(44/35/1.22)
HS111	10/3	57(73/ 76/1.52)	77(124/128/1,82)
HSH2	10/4	31(55/ 31/0.98)	31(55/ 31/0.99)
HS113	10/6	29(36/ 41/0.99)	27(32/ 38/0.94)

Table 2: Number of iterations (No. functions/No. gradients/CPU sees.) for convergence of several Hock and Schittkowski problems.

Note that the multipler free method performs well with respect to the original reduced Hessian method. It requires only half the iterations on problems HS 102 and HS 103, although it requires significantly more iterations for problems HS 101 and HS 111* Otherwise the performance of both algorithms is quite similar.

Table 3 presents the results for somewhat larger examples from the

CUTE collection. The comments made prior to the Hock and Schittkowski problems also apply here. The problems are again solved within a tolerance of 10^{-5} and, in addition to the number of iterations required for convergence the CPU times on a DEC ALPHA 3000-400 are also reported.

The first two problem sets in Table 3, EIGENC2 and EIGENCCO, are problems with quadratic objective functions and quadratic constraints that solve symmetric eigenvalue problems as a system of nonlinear equations. Many of the cases reported in Table 3 require a change of basis to avoid poorly conditioned bases. The problems are initialized at a point which satisfies the equality constraints. The ORTHREGA, ORTHREGC and ORTHREGD problems are orthogonal regressions where the objective is to fit orthogonally a cardioid to a set of point in the plane [15]. These problems are initialized at a point where the objective function and its gradients are uniquely zero. This causes the initial null space move and Lagrange multipliers to be zero, and the initial value of the penalty parameter is set to one.

In comparing both algorithms, each has its advantages on selected problems and the overall performance of both methods is similar. One advantage of the multiplier free method is that the lower bound on the penalty parameter leads to less severe penalties on the constraint violations and often allows larger steps to be taken in the linesearch. This can be seen, for instance, in the larger ORTHREGA problems and in some of the ORTHREGC problems.

4.2 Process Optimization Case Study

In order to illustrate the tailored reduced Hessian method described above, we summarize the study reported in [23] for the optimization of the operating conditions of a typical chemical process. Here we consider the Hydrodealkylation (HDA) process illustrated in Figure 1 which is used to manufacture nitration-grade toluene by the thermal dealkylation of nitration-grade benzene. This high temperature, noncatalytic process converts toluene to benzene in the presence of excess hydrogen; the only byproducts produced in any significant quantity are methane and diphenyl. Once the reactor effluent has been cooled, and the bulk of the light components separated via a flash unit, the liquid stream is sent to the distillation train. Both unreacted hydrogen and unreacted toluene are recycled. Further details on this process may be found in **[18]** and in [12]. While these authors discuss the grass-roots design of the plant, we focus on the real-time optimization of the operating conditions of the process given an existing design; the calculations are based

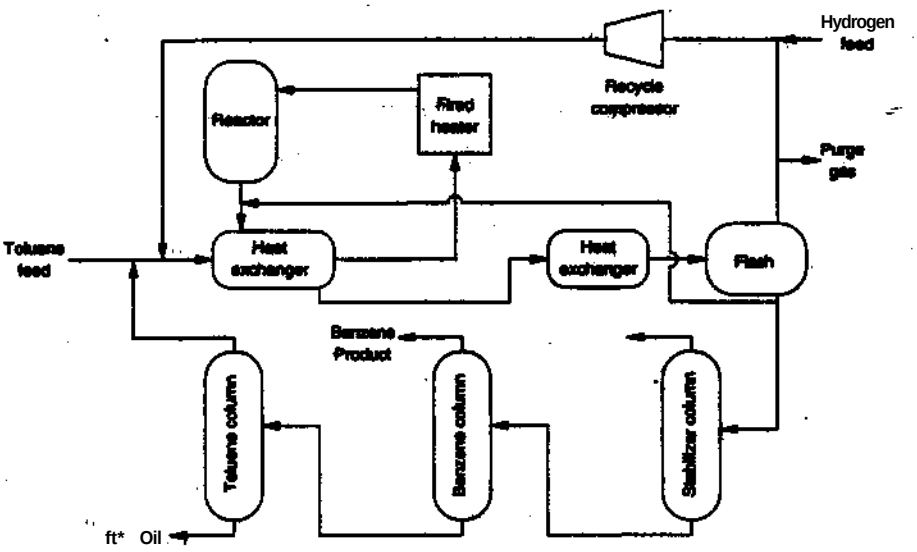

Figure 1: Simplified flowsheet of HDA process.

on a fixed feed rate of toluene.

The equation-based model for the HDA process includes both mass and energy balances for each of the units. The design equation for the reactor, which is modeled as a PFR, can be integrated to give a closed form expression if an average reactor temperature is used. The stablilizer column is approximated by a sharp split into light components (hydrogen and methane) and heavy components (benzene, toluene and diphenyl). The benzene and toluene columns, on the other hand, are treated as "special" units. Here, the distillation equations are obtained from the Naphthali-Sandholm model, UNIDIST, which is part of the SEPSIM process simulator [2]. This existing package incorporates an efficient Thomas algorithm which exploits the block tridiagonal structure of the distillation equations. Design constraints include the purity of the benzene product (> 99.7%) as well as upper bounds on the amount of fresh hydrogen available and the reactor effluent temperature. The reduced Hessian SQP algorithm also requires first order derivatives of the model with respect to all the variables. The derivatives of the distillation equations with respect to the internal variables are available analytically within UNIDIST; the derivatives with respect to the independent variables (distillate rate and reflux ratio) and the input streams involved in the distillation equations were calculated using finite differences.

For the remainder of the model, analytic expressions for all the derivatives were generated.

Here, we consider three cases for the optimization of the HDA process:

1. the toluene column modeled using UNIDIST and the benzene column modeled as a sharp split,
2. the benzene column modeled using UNIDIST and the toluene column modeled as a sharp split and
3. two benzene columns in parallel modeled using UNIDIST and the toluene column modeled as a sharp split.

In addition, we use three solution strategies:

1. a simultaneous modular approach; here only the reflux ratio and the distillate rate are included in the optimization problem and the distillation equations are fully converged at every iteration using the Thomas algorithm within UNIDIST,
2. an equation oriented approach where the UNIDIST package has been significantly modified so that the Jacobian elements for the distillation equations can be passed to a sparse linear equation solver (we use MA28 from the Harwell library) at every iteration and
3. the tailored integrated approach where UNIDIST is only slightly modified such that the Newton step and the projected gradients $(C^{T}N)$ for the distillation equations are calculated via the Thomas algorithm and then collected and passed to the reduced Hessian method. A complete description of the tailored decomposition strategy is given in [23].

The last case is of practical interest when, for example, benzene products of different purity are required. The main reason for solving this case though is to have a large number of internal or 'hidden' variables and equations. For all three strategies, the problem is initialized at the same starting point. The objective function to be minimized is $\langle f \rangle$ = *(reflux ratio)* - *(distillate* note). The results in Table 4 give the number of iterations and CPU time on a DEC 5000/200 for convergence (Kuhn-Tucker error < 10^{-4}). For each problem, all the methods were initialized at the same point and converged to the same solution.

The results in Table 4 show that for this larger system, the difference in performance between the simultaneous modular and the integrated approaches is significant; while the number of SQP iterations is approximately

the same, the CPU time differs by almost an order of magnitude. Also, the number of iterations required by the equation oriented and the tailored approaches is identical and the tailored method is slightly more efficient than the equation oriented approach. More importantly, though, using specialized solution procedures to generate the Newton step for individual units also predetermines the pivot sequence for these units. For existing unit models which have been tested on a wide range of problems, these pivot sequences are often known to be very robust with few failures due to badly conditioned or singular Jacobian matrices.

5 Conclusions

Process optimization problems frequently incorporate nonlinear models that can be solved reliably and efficiently by specialized, Newton-based procedures. These procedures take advantage of the equation structure and allow for specialized matrix decomposition algorithms. The aim of this paper is to study an optimization strategy that uses existing process models in a simultaneous convergence scheme. The SQP algorithm developed in section 2 requires only a Newton step (p_Y) from the model equations and 'sensitivity' of this step with respect to the decision variables $(C^{-1}N)$. This information is relatively easy to obtain without modification of the model solution procedure or its data structures. Moreover, if the degrees of freedom are small, only a few additional backsolves are required for $(C^{-1}N)$.

One restriction to the use of existing procedures is that the matrix C or its LU factors may be difficult to access. As a result, a multiplier free SQP approach was developed with relatively few modifications of the reduced Hessian method analyzed in [4]. The main differences are due to the estimates for the penalty parameter for the line search function and in the calculation of quasi-Newton updates for the reduced Hessian and the cross term approximations. In section 3 we summarize the convergence properties of this method and show that the desired global and 1-step superlinear convergence properties are retained from those in [4]. Simple QP extensions are also derived in section 2 to deal with bound constrained problems. Moreover, with the addition of the QP step the multiplier free algorithm can easily be extended to include trust region concepts. This will be the focus of our future work.

Finally, numerical experiments for this approach indicate no loss of efficiency or reliability over the method described in [4]. On the other hand, the

multiplier free approach allows an easy integration with specialized Newton-based equation solvers, in order to extend them to deal with optimization problems. This was demonstrated for flowsheet optimization through the use of block tridiagonal distillation models. Recent studies [24], [25] also describe the integration of the multiplier free approach to the optimization of systems described by boundary value problems (BVPs). In this case, the BVP solver COLDAE [l] was combined with the multiplier free method to solve problems in parameter estimation, optimal control and reactor design. Implementation was relatively straightforward as all of the data structures, linear algebra and solution procedures were preserved. Moreover, the model specification routines within COLDAE could be used directly to set up the optimization poblems.

As a result of these efforts, we believe that the multiplier free method has good potential for solving large optimization problems with few degrees of freedom. The additional benefit of this approach is that existing structured solution procedures can be exploited for these problems.

References

[1] Ascher, U. and R. Spiteri, Collocation software for boundary value differential-algebraic equations, SIAM J. Scient. Stat. Comput., to appear (1995)

[2] Andersen, P.M., F. Genovese and J. Perregard Manual for Steady State Simulator, SEPSIM. Institut for Kemiteknik, DTH, Lyngby, Denmark (1991).

[3] Biegler, L. T. *Convergence analysis for the multiplier free reduced Hessian method,* EDRC Report (1995)

[4] Biegler, L. T., J. Nocedal and C¥ Schmid, *A reduced Hessian method for large-scale constrained optimization,* SIAM J. Opt., 5, 2, (1995), pp. 314-347

[5] Biegler, L. T., J. Nocedal and C. Schmid, *Numerical experience with a reduced Hessian method for optimization,* in preparation, (1995).

[6] Bongartz, I., A. R. Conn, N. Gould, and P. L. Toint, CUTE: Constrained and Unconstrained Testing Environment (1993)

[7] Byrd, R. H., and J. Nocedal, *A tool for the analysis of quasi-Newton methods with application to unconstrained minimization,* SIAM J. Numer. Anal., 26 (1989), pp. 727-739.

[8] Byrd, R. H., and J. Nocedal, *An analysis of reduced Hessian methods for constrained optimization,* Math. Programming, 49 (1991), pp. 285-323.

[9] Chamberlain, R., C. Lemarechal H. C. Pedersen and M. J. D. Powell, *The watchdog technique, for forcing convergence in algorithms for constrained optimization, Math.* Programming Studies, 16 (1982), pp. 1-17.

[10] Conn, A. R., *Constrained optimization using a nondifferentiable penalty function,* SIAM J. Num. Anal., 13 (1973), p£. 145-154.

[11] Dennis, J. E., and R.B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations,* Prentice-Hall, Inc., Englewood Cliffc, NJ, 1983.

[12] Douglas, J.M. *Conceptual Design of Chemical Processes.* McGraw Hill (1988).

[13] Fletcher, R, *Practical Methods of Optimization* (second edition), John Wiley and Sons, Chichester, 1987.

[14] Goldfard, D. and A. Idnani, A numerically stable dual method for solvr ing strictly convex quadratic programs, *Math. Programming,* 27, p. 1 (1983)

[15] Gulliksson, M., Algorithms for nonlinear least squares with applications to orthogonal regression, UMINF-178.90, University of Umea, Sweden (1990)

[16] Han, S. *P.A globally convergent method for nonlinear programming,* Journal on Optimization Theory and Application, 22/3 (1977), pp. 297-309.

[17] Hock, W., and K. Schittkowski, Test examples for nonlinear programming codes, Lecture notes in economics and mathematical systems 187, Springer-Verlag, Berlin (1981)

[18] McKetta, J.J. (ed.), *Encyclopedia of Chemical Processing and Design.* vol 4, Dekker, New York, 1977, p. 182. (1977)

[19] Murtagh, B. and M. Saunders, MINOS User's Guide, Report SOL 83-20R (1983)

[20] J. Nocedal and M. L. Overton, *Projected Hessian updating algorithms for nonlinearly constrained optimization,* SIAM Journal on Numerical Analysis, 22 (1985), pp. 821-850.

[21] Sargent, R. W. H., Survey of SQP methods, this workshop (1995)

[22] Schmid, C. and L.T. Biegler, Quadratic Programming Methods for Tailored Reduced Hessian SQP. *Computers and Chemical Engineering,* 18/9, pp. 817-832 (1994)

[23] Schmid, C. and L.T. Biegler, A Simultaneous Approach for Flowsheet Optimization with Existing Modeling Procedures, *Turns. I. Chem. Eng.*, 72A, pp. 382-388 (1994)

[24] Tanartkit, P. and L. T. Biegler, "Stable Decomposition for Dynamic Optimization," I & EC Research, 34, p. 1253 (1995)

[25] Tanartkit, P., and L. T. Biegler, "Reformulating Hl-Conditioned DAE Optimization Problems," submitted for publication (1995)

N/M	With Multipliers	Multiplier Free
EIGENC2	Quadratic constraints	
	MA28 used to select dependent variables, scaled	
30/15	32(59/39/1.06)	29(**49/34/0.99)**
56/28	58(116/72/2.30)	44(90/56/1.99)
90/45	67(125/84/3.92)	**67(124/82/3.68)**
EIGENCCO	Quadratic constraints	
	MA28 used to select dependent variables, scaled	
30/15	33(57/ 41/ 1.13)	30(50/37/1.08)
56/28	45(81/ 56/ 2.23)	63(125/ 84/ 2.83)
90/45	65(122/ 88/ 4.84)	56(89/78/4.24) -
ORTHREGA	Quadratic Constraints	
	MA28 used to select dependent variables, scaled	
13/4	2(1/ 2/ 0.31)	2(1/ 2/ 0.31)
37/16	91(189/ 97/ 3.39)	104(240/113/ 3.75)
133/64	308(608/322/ 74.iO)	136(286/155/ 34.77)
517/256	298(681/338/1385.73)	197(419/220/971.31)
ORTHREGC	Quadratic constraints	
	larger of [x,y] selected ;as dependent variables, scaled	
205/100	49(84/ 65/ 20.47)	42(59/ 55/ 18.00)
405/200	123(181/182/ 210.44)	69(97/ 91/ 125.48)
505/250	107(185/170/ 268.22)	129(257/208/ 324.08)
ORTHREGD	Quadratic constraints	
	MA28 used to select dependent variables, scaled	
23/10	25(30/40/0.73)	22(27/ 39/ 0.69)
103/50	29(38/ 48/ 4.30)	24(29/ 35/ 3.53)
203/100	23(27/ 37/ 11.21)	25(29/38/12.13)
303/150	33(41/ 55/ 33.45)	25(28/ 37/24.69)

Table 3: Number of iterations (No. functions/No. gradients/CPU sees.) for several problems from the CUTE collection

Table 4: Results for the HDA process examples

	Toluene column	Benzene column	Two columns
Components	2	3	3 and 3
TVays	12	30	30 and 30
Number of variables			
decisions	7	7	9
flowsheet	184	184	202
internal (methods (2) and (3))	24	90	180
Number of equality constraints			
method (1)	184	184	202
methods (2) and (3)	208	274	382
Number of nonzero Jacobian elements			
method (1)	486	495	549
methods (2) and (3)	654	1431	2421
(1) Simultaneous modular approach			
Iterations (SQP/Newton)	9/184	11/1243	12/2698
CPU Time (s)	10.1	29.3	62.2
(2) Equation oriented approach			
Iterations (SQP)	7	14	12
CPU Time (s)	1.9	5.1	7.8
(3) Tailored approach			
Iterations (SQP)	7	14	12
CPU Time (s)	1.9	5.0	7.4